ИНФОРМАТИКА

2014 **январь-март** **№ 1**

УДК 004

Б.А. Залесский, Э.Н. Середин

РЕАЛИЗАЦИЯ АЛГОРИТМОВ ОТСЛЕЖИВАНИЯ ОБЪЕКТОВ НА ОСНОВЕ ТЕХНОЛОГИИ CUDA

Предлагаются быстрые версии корреляционных алгоритмов отслеживания объектов на видеопоследовательностях, снятых нестабилизированной камерой. Алгоритмы предназначаются для реализации на CUDA. Они основываются на сравнении корреляции области изображения, содержащей объект интереса, с соответствующими ей по размеру областями текущего кадра видеопоследовательности вместе с анализом межкадровой информации. Применение технологии программирования видеокарты CUDA позволило добиться выполнения алгоритмов в режиме реального времени. Для повышения точности и устойчивости предложенных версий алгоритмов используется робастная версия фильтра Калмана. Показывается работоспособность предложенных версий даже в случае видеопоследовательностей, снятых нестабилизированной камерой.

Введение

За последние годы алгоритмы отслеживания объектов получили широкое применение в различных сферах деятельности общества, включая охрану окружающей среды, картографию, навигацию, безопасность движения и т. д.

Для отслеживания объектов на видеопоследовательностях, снятых в разных условиях, используются различные алгоритмы, поэтому рассмотрим случай, когда видеопоследовательность снята движущейся нестабилизированной камерой, например, установленной на борту летательного аппарата. Задача состоит в том, чтобы отследить в реальном времени объект на кадрах видеопоследовательности. Сложность данного случая заключается в невозможности предсказать заранее положение объекта (и, следовательно, ограничить его область поиска) на текущем кадре, а также в том, что обычно при такой съемке на кадрах присутствуют шумы разных типов.

В большом числе статей по данной тематике содержится множество разнообразных алгоритмов, предназначенных для решения сформулированной задачи. Среди них алгоритмы, основанные на сравнении гистограмм яркостей и ориентированных градиентов, оптических потоков, корреляций, текстур, активных контуров и др. [1–6]. Одна из главных трудностей, общая для всех известных подходов, – малое время, имеющееся для оценки положения объекта на текущем кадре, обычно ограниченное 25÷40 мс. Некоторые надежные и точные алгоритмы невозможно выполнить на компьютере за столь короткое время. До недавнего времени это касалось и корреляционных алгоритмов, достаточно надежных и точных, но, к сожалению, требующих вычисления большого объема операций. Непосредственное оценивание координат объекта, отслеживаемого на видеопоследовательности стандартного размера с помощью корреляции Пирсона (при поиске по всему кадру), требует нескольких минут вычислений на современном процессоре. Даже использование быстрого преобразования Фурье и SSE-инструкций для подсчета корреляции не позволяет достичь реального времени сопровождения.

Однако ситуация изменилась с появлением технологии программирования CUDA, позволяющей производить параллельные вычисления на видеокартах, выпускаемых компанией Nvidia. Новые архитектуры технологии CUDA, такие как Fermi и Kepler, позволяют выполнить корреляционные алгоритмы отслеживания объектов движущейся нестабилизированной видеокамерой в режиме реального времени на видеокартах GeForce шестой серии и выше.

В статье описаны такого вида реализации вместе со сравнением корреляционного подхода к решению задачи отслеживания с некоторыми известными подходами. Представлено также описание робастной версии фильтра Калмана, использованной для повышения точности и стабильности трекинга.

1. Краткое описание алгоритма отслеживания

Первоначально для решения задачи отслеживания объектов на видеопоследовательностях, снятых движущейся нестабилизированной видеокамерой, были реализованы и протестированы на реальных данных (рис. 1) несколько известных алгоритмов, позволяющих решать задачу в режиме реального времени, и неоптимизированные корреляционные алгоритмы в тестовом режиме. Корреляционные алгоритмы оказались более точными и надежными. Для повышения надежности и точности сопровождения была использована робастная версия фильтра Калмана.

Рис. 1. Пример кадра видеопоследовательности, полученной с борта самолета

Обозначим через $\mathbf{I} - m \times n$ полутоновое или RGB-изображение с пикселами $p = (x, y)$ и яркостями I_p или цветом $(I_{R,p}, I_{G,p}, I_{B,p})$. Кадр видеопоследовательности в момент времени $t \in \{0, 1, ...\}$ обозначим $\mathbf{I}_t$.

Процесс сопровождения начинается с выделения объекта интереса. Пусть $\mathbf{O}_0(p_{obj})$ – $k \times k$-часть начального кадра $\mathbf{I}_0$, содержащая отслеживаемый объект, с центром в пикселе p_{obj}. Пусть $\mathbf{O}_t(p, \alpha)$ – часть текущего кадра $\mathbf{I}_t$, повернутого на угол α. Она представляет собой $k \times k$-квадрат со сторонами, параллельными сторонам неповернутого кадра, и центром p относительно исходного неповернутого изображения.

Замечание. В данной версии алгоритма не описывается применение масштабной пирамиды, позволяющей вести слежение за объектом при изменении высоты видеокамеры, так как проведенные эксперименты показали, что в рассматриваемом случае в каждый момент времени достаточно использовать только три уровня масштабной пирамиды: актуальный уровень предыдущего кадра и ближайшие к нему верхний и нижние уровни.

Два типа корреляции были использованы для оценки положения объекта:

$$r_1(p, \alpha) = r_1(\mathbf{O}_0, \mathbf{O}_t(p, \alpha)) = \frac{1}{k^2} \sum_{x,y=0}^{k} O_{0,x,y} O_{t,x,y}(p, \alpha) - \bar{O}_0 \bar{O}_t(p, \alpha),$$

где $\bar{O}_0$ и $\bar{O}_t(p, \alpha)$ – средние значения изображений $\mathbf{O}_0$ и $\mathbf{O}_t(p, \alpha)$, и

$$r_2(p,\alpha) = r_2(\mathbf{O}_0, \mathbf{O}_t(p,\alpha)) = \frac{r_1(p,\alpha)}{SSQ(\mathbf{O}_0)SSQ(\mathbf{O}_t(p,\alpha))},$$

где $SSQ(\mathbf{I})$ – стандартное уклонение яркостей (или цвета) изображения $\mathbf{I}$.

Оценки положения p_t отслеживаемого объекта в момент времени t вычисляются как

$$\hat{p}_{1,t} = \arg\max_{p,\alpha} r_1(p,\alpha) \text{ и } \hat{p}_{2,t} = \arg\max_{p,\alpha} r_2(p,\alpha).$$

Максимум берется по всем пикселам изображения $\mathbf{I}$ и всем допустимым углам α.

Данные оценки хорошо известны не только в обработке изображений, поэтому их использование не является новшеством. Новизна заключается, во-первых, в быстрой программной реализации оценок $\hat{p}_{1,t}$ $\hat{p}_{2,t}$, основанной на применении технологии программирования видеокарты CUDA, которая сделала возможным выполнение алгоритма в режиме реального времени, и, во-вторых, в использовании робастной версии фильтра Калмана, которая сделала процесс отслеживания более устойчивым и точным. Ключевые моменты аппаратной оптимизации предложенного подхода приведены в следующем разделе.

Опишем кратко применяемую робастную версию фильтра Калмана. Традиционно схема линейного фильтра Калмана без управления задается следующими уравнениями:

$$\begin{aligned} \hat{x}_t^- &= F_{k-1}\hat{x}_{t-1}^+; \\ \hat{x}_t^+ &= \hat{x}_t^- + K_t(y_t - H_t\hat{x}_t^-), \end{aligned} \tag{1}$$

где $\hat{x}_t^-$ – предсказание (априорная оценка); $\hat{x}_t^+$ – апостериорная оценка; y_t – входной зашумленный сигнал. Назначение и вид матриц F_{k-1}, K_t и H_t приведены в деталях в [7].

При использовании классической версии фильтра Калмана для сглаживания и стабилизации оценок $\hat{p}_{1,t}$ $\hat{p}_{2,t}$ возникают грубые погрешности в случаях, когда оценка $\hat{p}_{i,t}$ указывает на ошибочное положение объекта на текущем кадре $\mathbf{I}_t$. Погрешности можно уменьшить следующим образом: для квадрата пикселов $S(\hat{x}_{t-1})$ с центром в $\hat{x}_{t-1}$ заранее фиксированного размера проверить условие попадания $\hat{p}_{i,t}$ в $S(\hat{x}_{t-1})$ и выбрать в качестве входного значения

$$y_t = \begin{cases} \hat{p}_{i,t}, \text{если } \hat{p}_{i,t} \in S(\hat{x}_{t-1}); \\ \hat{x}_t^- \text{ в противном случае.} \end{cases}$$

Кроме того, для заранее выбранного целого числа N_{out} следует вычислить, сколько последовательных оценок $\hat{p}_{i,t}$ оказалось вне $S(\hat{x}_t^-)$. Как только число таких критических оценок станет больше чем N_{out}, следует инициализировать новый фильтр Калмана следующим образом:

1) в качестве решения положить $\hat{x}_t^+ = \hat{p}_{i,t}$;

2) для некоторого заранее выбранного целого числа N_{in} проверить, превосходит ли количество последовательных оценок $\hat{p}_{i,t+k}$, оказавшихся в квадрате $S(\hat{x}_t^+)$, число N_{in}. Если превосходит, использовать описанную выше робастную версию фильтра Калмана, иначе повторить п. 1.

Общая схема работы предложенного алгоритма отслеживания объектов на видеопоследовательностях, снятых нестабилизированной видеокамерой, показана на рис. 2.

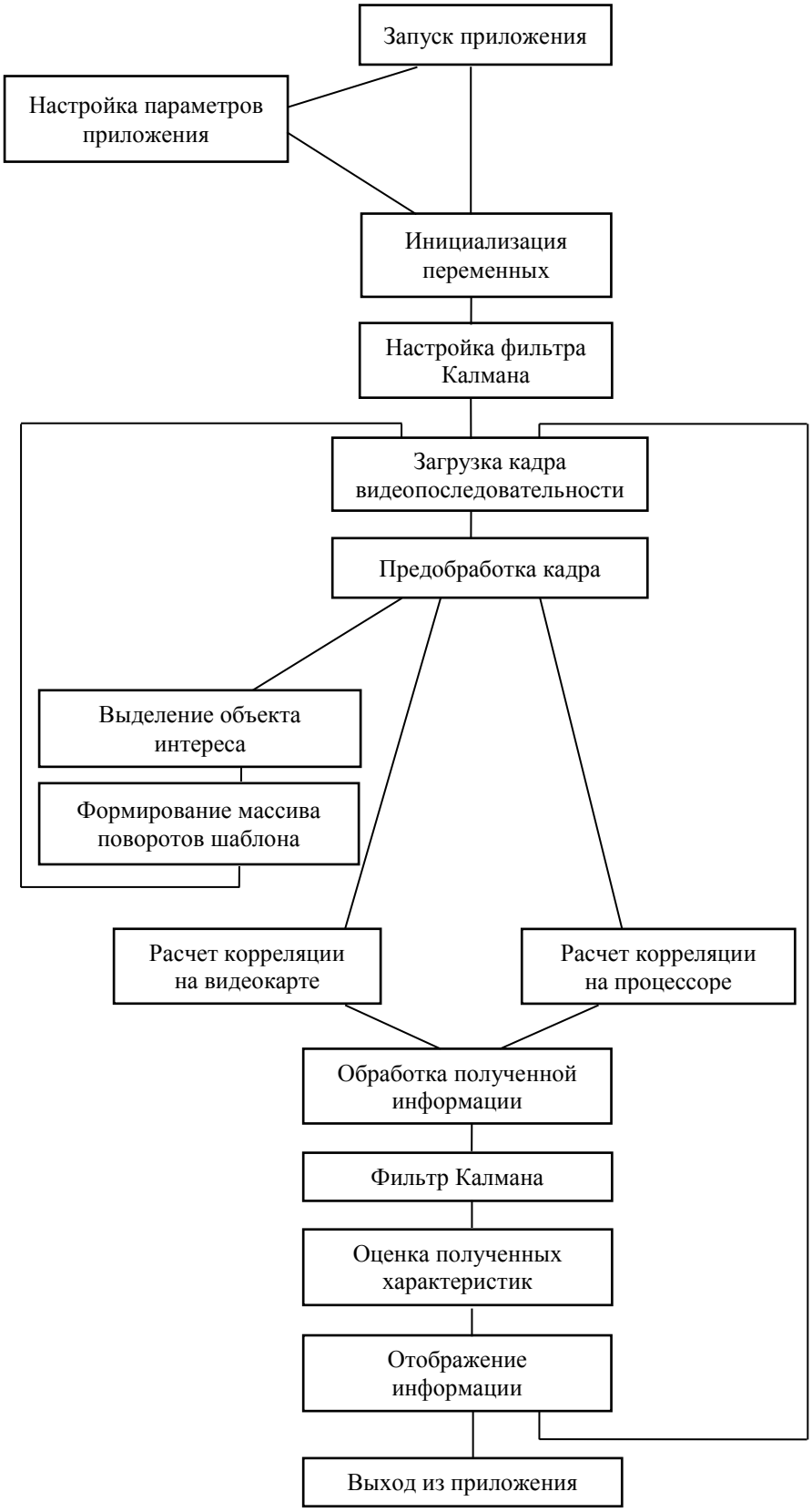

Рис. 2. Общая схема работы приложения по отслеживанию объектов

2. Особенности реализации алгоритма

Корреляционные алгоритмы требуют большого числа вычислений, поэтому их применение в задачах отслеживания объектов до недавнего времени было ограничено из-за невозможности выполнения в режиме реального времени. Для преодоления этого недостатка была использована технология программирования CUDA, которая обеспечила необходимый прирост производительности вычислений за счет параллельной обработки данных.

Реализация на CUDA корреляционных алгоритмов, предназначенных для отслеживания объектов на видеопоследовательностях в режиме реального времени, – нетривиальная задача. Для ее решения первоначально был формализован в терминах технологии CUDA последовательный алгоритм вычисления большого числа корреляций, необходимых для отслеживания объекта на текущем кадре. Затем на каждом шаге построения параллельной версии алгоритма выделялись его критические блоки, вычисление которых занимало бóльшую часть времени. Производилось распараллеливание и оптимизация выполнения выделенных блоков до уровня, позволяющего устранить их критичность для достижения режима реального времени. После получения версии алгоритма, допускающей выполнение в режиме реального времени, были распараллелены оставшиеся блоки алгоритма для достижения максимального быстродействия построенной версии.

Практическая реализация алгоритма показала, что наилучшее быстродействие достигается при использовании 2D-текстур и разделяемой памяти. Одной из особенностей работы с текстурами является то, что на видеокарте они доступны в режиме для чтения, а запись в них может производить только процессор.

Эффективность использования текстур проявляется в том, что работа с данными в них происходит через текстурные выборки, содержание которых кешируется. Использование разделяемой памяти объясняется тем, что она является самой быстрой памятью на видеокарте наравне с регистрами и может использоваться для хранения промежуточных вычислений. Расчет на видеокарте всей совокупности корреляций, необходимой для отслеживания объекта интереса на текущем кадре, состоит из следующих этапов:

– инициализации данных, необходимых для запуска ядра на видеокарте;

– заполнения текстуры **source** данными из кадра видеопоследовательности, на котором будет происходить поиск объекта;

– заполнения **n** текстур **template** данными из массива, в котором находятся **n** вариантов поворота объекта поиска;

– выделения памяти на видеокарте под массив выходных значений;

– запуска ядра на видеокарте для выполнения вычислений с использованием специально разработанной топологии взаимодействия блоков и нитей (определение этих терминов дается в [8, 9]), в которой каждому пикселу текстуры **source** ставится в соответствие отдельный блок. Количество нитей для блока выбирается исходя из размеров текстуры **template** для обеспечения наиболее быстрого прохода по ячейкам текстуры;

– создания переменных для индексации внутри видеокарты;

– проверки на наличие выхода переменных за пределы допустимых диапазонов значений;

– запуска **n** раз для каждого пиксела текстуры **source** функции **f()**, вычисляющей корреляцию между областью изображения с центром в данном пикселе, размер которой соответствует размеру окна, выделенного оператором на исходном кадре, и каждой текстурой **template**, где **n** – количество заданных поворотов объекта поиска (рис. 3);

– занесения результатов в выходной массив, находящийся в глобальной памяти видеокарты, с учетом предложенной авторами индексации внутри видеокарты, что позволяет исключить конфликты доступа к ячейкам данного массива;

– копирования выходного массива, в котором содержатся значения корреляции для каждого пиксела текущего кадра видеопоследовательности и для каждого заданного угла поворота, из памяти видеокарты в оперативную память.

Быстрая реализация функции **f()** может быть представлена в виде следующих последовательно выполняющихся шагов:

– считывания нитями блока в параллельном режиме данных из текстуры **source** и текстуры **template** в регистры видеокарты для выполнения арифметических операций, необходимых для вычисления корреляций;

– выполнения в параллельном режиме операций, необходимых для вычисления скалярных произведений и средних значений для расчета корреляций $r_1(p,\alpha)$ и $r_2(p,\alpha)$, с использованием специализированных математических функций технологии CUDA;

– занесения результатов вычислений в разделяемую память;

– использования при больших размерах текстуры **template** для ускорения вычислений приема работы с разделяемой памятью, называемого редукцией [9];

– непосредственного вычисления корреляций $r_1(p,\alpha)$ и $r_2(p,\alpha)$;

– возвращения из функции значения рассчитанных корреляций $r_1(p,\alpha)$ и $r_2(p,\alpha)$.

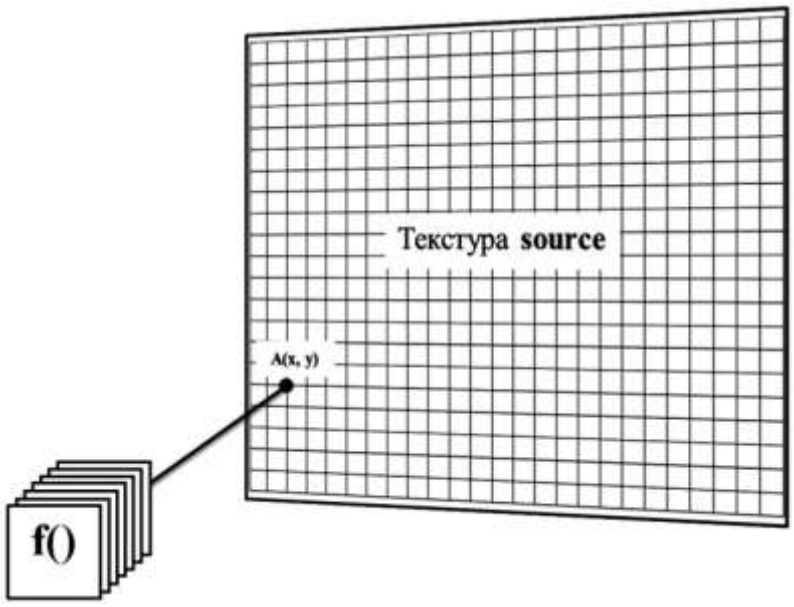

Рис. 3. Пример расчета корреляции на видеокарте в точке **A** с координатами **(x, y)**

Укажем некоторые особенности программной реализации алгоритма. Функция **f()**, вычисляющая корреляции $r_1(p,\alpha)$ и $r_2(p,\alpha)$, написана на расширенном языке CUDA C без использования сторонних библиотек и является платформенно-независимой. Для операционных систем Windows созданы отдельные 32- и 64-битные библиотеки в виде файлов *.dll, которые могут быть использованы в любых языках программирования.

Размер текстуры **source** может быть произвольным, а размер текстуры **template** желательно, чтобы был кратным двойке (8x8, 16x16, 32x32, 64x64, 128x128). При заполнении текстур **template** используются заранее подготовленные таблицы соответствия пикселов исходной и повернутой текстур, позволяющие поворачивать шаблон на произвольный угол без каких-либо расчетов и интерполяции.

В описываемом алгоритме не используется масштабирование изображений при поиске объекта. В замечании уже упоминалось, что добавление двух дополнительных уровней масштабной пирамиды – верхнего и нижнего – и расчет корреляции на них не вносят принципиальных трудностей, так как дополнительные вычисления на видеокарте могут проводиться параллельно.

3. Результаты тестирования алгоритма

Тестирование алгоритма проводилось на реальных, снятых с борта летательного аппарата, и модельных видеопоследовательностях. Результаты экспериментов приведены в табл. 1.

Приблизительно 90–95 % времени обработки одного кадра видеопоследовательности тратится на расчеты, которые производятся на видеокарте, а остальная часть – на загрузку изображения, предварительную обработку и т. д.

Таблица 1

Время обработки одного кадра видеопоследовательности, мс (шаблон 8x8 пикселов)

Устройство	Разрешение изображения, пикселов						
	256x256	640x480	768x576	1024x768	1440x1080	1920x1080	4000x4000
GPU GTX 260	14	44	58	106	205	271	2069
GPU GTX 550Ti	9	35	46	86	169	225	1730
GPU GTX 650Ti	7	22	29	53	104	138	1049

В табл. 1 время расчетов одного кадра видеопоследовательности указано для коэффициента масштабирования, равного 1, хотя в разработанном алгоритме применяются и другие коэффициенты масштабирования – 4, 16. Как показало тестирование, масштабирование не сказывается на точности работы алгоритма. При масштабировании одного кадра видеопоследовательности время расчетов уменьшается кратно коэффициенту сжатия, что позволяет использовать видеопоследовательности с более высокими разрешениями. Еще одной особенностью реализации алгоритма с помощью технологии CUDA является то, что при увеличении размера шаблона время работы алгоритма растет незначительно. Например, при увеличении размера шаблона в 16 раз время обработки одного кадра возрастает на 20–25 % в отличие от процессорной версии алгоритма, время работы которой возрастает в 16 раз.

Для сравнения точности разработанного алгоритма выбраны другие алгоритмы поиска объекта интереса на кадрах видеопоследовательности, работающие в режиме реального времени и основанные на сравнении гистограмм ориентированного градиента [2, 10] и текстурных признаков областей изображений [1]. В табл. 2 приведены проценты правильного нахождения ранее заданного объекта интереса на кадрах видеопоследовательности плохого качества. Под правильным понимается нахождение такой области изображения, которая содержит не менее 50 % площади искомого объекта интереса.

Таблица 2

Процент нахождения объекта интереса на кадрах видеопоследовательности

Алгоритм	Точность, %
Гистограмма ориентированного градиента	38
Текстура по трем точкам	60
Текстура по двум точкам	74
Описываемый в статье алгоритм	97

Из табл. 2 видно, что корреляционный алгоритм превосходит по точности другие протестированные алгоритмы на видеопоследовательностях низкого качества (пример кадра такой видеопоследовательности приведен на рис. 1).

Заключение

Представлено описание корреляционного алгоритма отслеживания объектов на кадрах видеопоследовательностей, полученных нестабилизированной камерой, и его быстрой программной реализации на CUDA, позволяющей проводить вычисления в режиме реального времени.

Приведены результаты сравнения на тестовых и реальных видеопоследовательностях точностей предложенного алгоритма с алгоритмами, основанными на сравнении гистограмм ориентированного градиента и текстурных признаков областей изображений. Точность предложенного корреляционного алгоритма оказалась выше, что соответствует результатам, полученным другими авторами.

Созданы 32- и 64-битные библиотеки для операционной системы Windows по расчету корреляций $r_1(p,\alpha)$ и $r_2(p,\alpha)$, реализованные с использованием технологии CUDA без каких-либо дополнительных сторонних библиотек.

В будущем планируется усовершенствовать алгоритм, включив в него шаги, связанные с быстрой сегментацией кадров видеопоследовательностей.

Список литературы

1. Yilmaz, A. Object tracking: A survey / A. Yilmaz, O. Javed, M. Shah // ACM Computing Surveys. – 2006. – Vol. 38, № 4.

2. Marimon, D. Orientation histogram-based matching for region tracking / D. Marimon, T. Ebrahimi // Proc. 8th Int. Workshop on Image Analysis for Multimedia Interactive Services WIAMIS. – Santorini, 2007. – P. 8–12.

3. Lowe, D. Object recognition from local scale invariant features / D. Lowe // Proc. Int. Conf. on Computer Vision ICCV. – Corfu, 1999. – P. 1150–1157.

4. Bay, H. Surf: Speeded up robust features / H. Bay, T. Tuytelaars, L. Van Gool // Proc. 9th Europ. Proc. 8th Int. on Computer Vision ECCV. – Graz, 2006. – P. 404–417.

5. Altmann, J. A Fast Correlation Method for Scale-and Translation-Invariant Pattern Recognition / J. Altmann, H.J. Reitböck // IEEE Trans. Pattern Anal. Mach. Intell. – 1984. – Vol. 6, № 1. – P. 46–57.

6. Object Tracking by Particle Filtering Techniques in Video Sequences / L. Mihaylova [et al.] // Advances and Challenges in Multisensor Data and Information. NATO Security Through Science Series. – Netherlands : IOS Press, 2007. – P. 260–268.

7. Simon. D. Optimal State Estimation. Kalman, H∞, and Nonlinear Approaches / D. Simon. – New Jersey : John Wiley & Sons, 2006. – 526 p.

8. Сандерс, Дж. Технология CUDA в примерах: введение в программирование графических процессоров / Дж. Сандерс, Э. Кэндрот. – М. : ДМК Пресс, 2011. – 232 с.

9. Боресков, А.В. Основы работы с технологией CUDA / А.В. Боресков, А.А. Харламов. – М. : ДМК Пресс, 2010. – 232 с.

10. Zalesky, B.A. Scale invariant algorithm to match regions on aero and satellite images / B.A. Zalesky, P.V. Lukashevich // Proc. 11th Int. Conf. on Pattern recognition and information processing PRIP 2011. – Minsk, 2011. – P. 25–31.

Поступила 3.12.2013

Объединенный институт проблем информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: eduard.seredin@tut.by

B.A. Zalesky, E.N. Seredin

IMPLEMENTATION OF OBJECT TRACKING ALGORITHMS ON THE BASIS OF CUDA TECHNOLOGY

A fast version of correlation algorithm to track objects on video-sequences made by a non-stabilized camcorder is presented. The algorithm is based on comparison of local correlations of the object image and regions of video-frames. The algorithm is implemented in programming technology CUDA. Application of CUDA allowed to attain real time execution of the algorithm. To improve its precision and stability, a robust version of the Kalman filter has been incorporated into the flowchart. Tests showed applicability of the algorithm to practical object tracking.